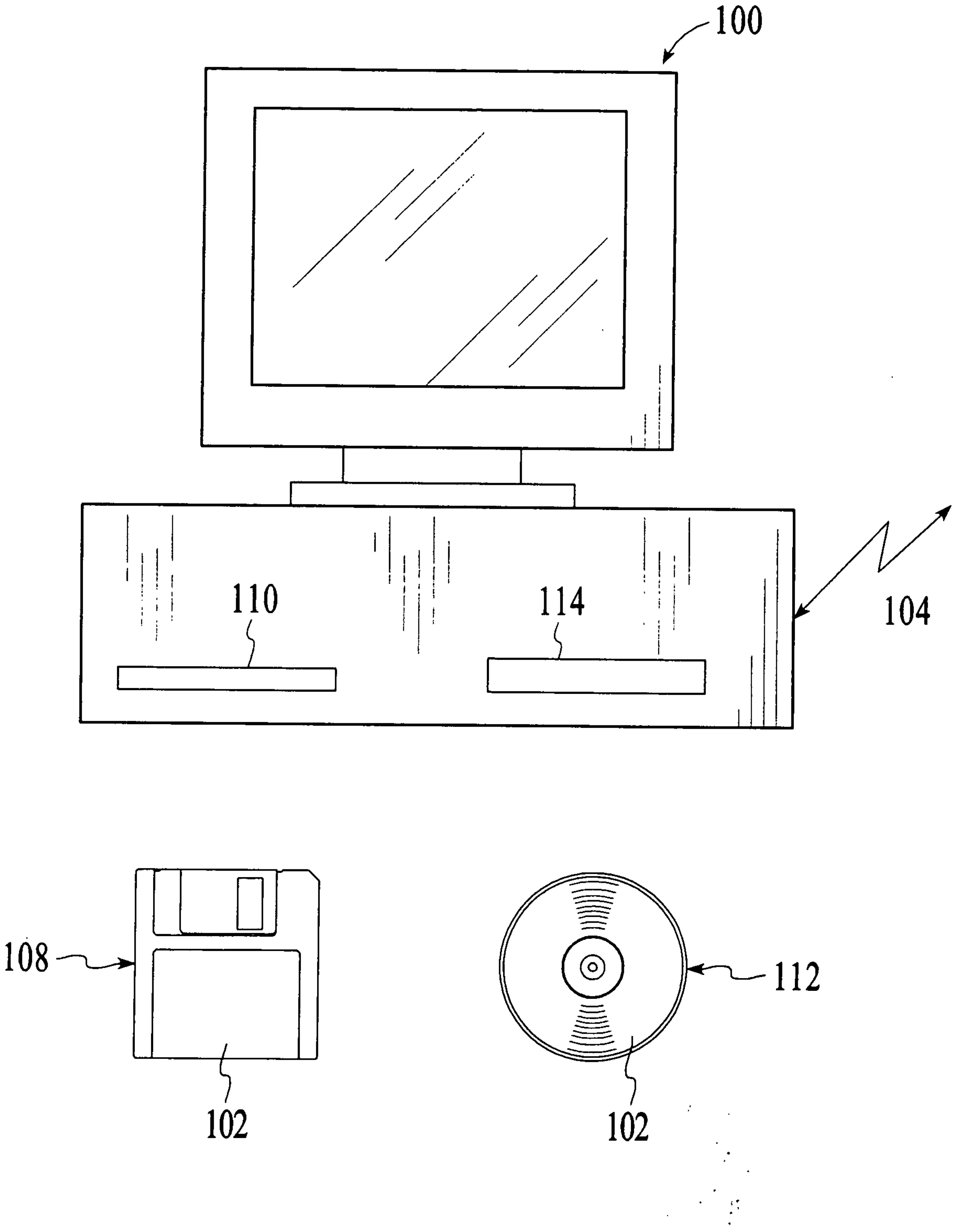
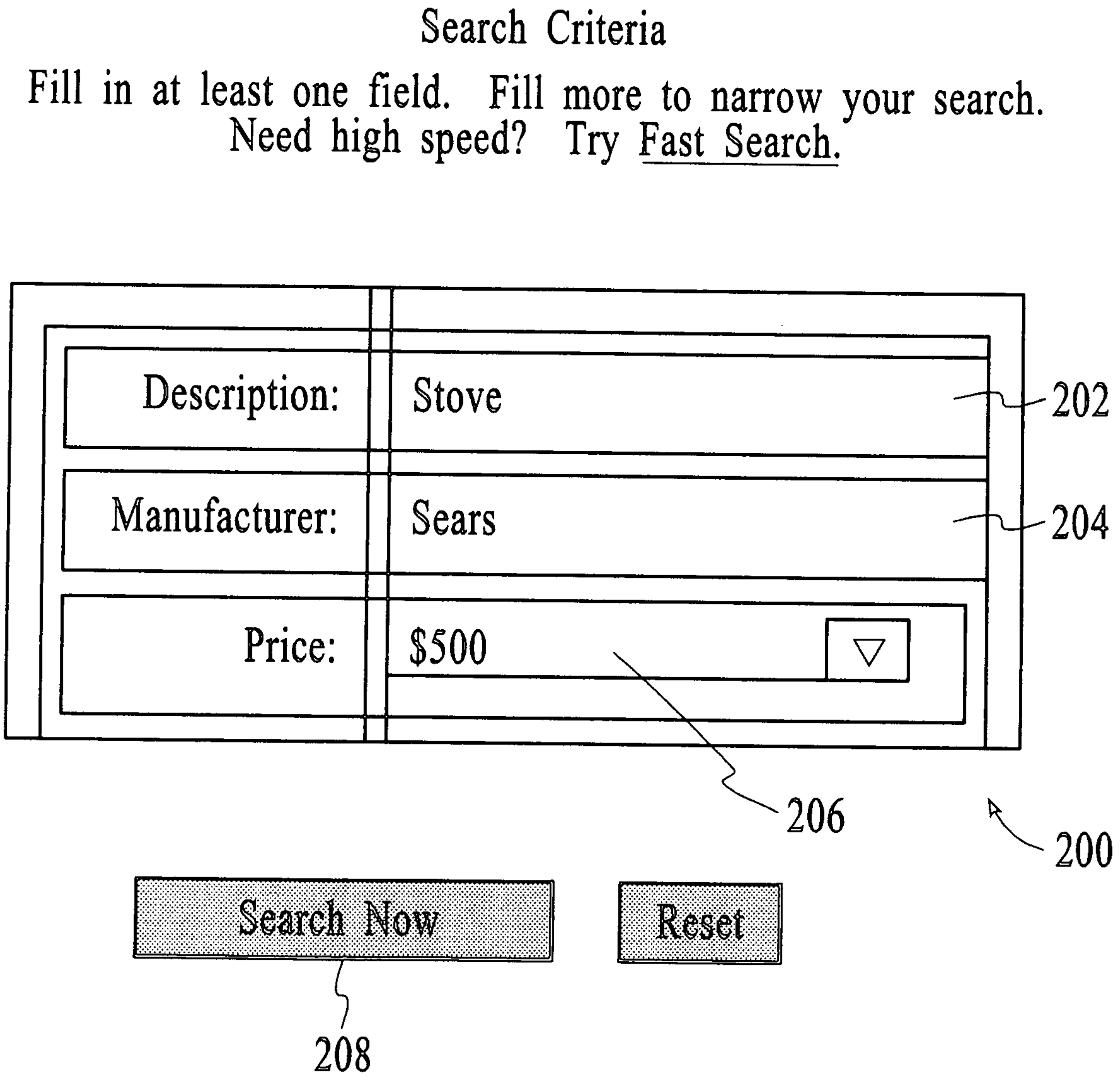


FIG. 1



Search Criteria

Fill in at least one field. Fill more to narrow your search.
Need high speed? Try Fast Search.

Description:	Stove
Manufacturer:	Sears
Price:	$500 ▽

FIG. 2

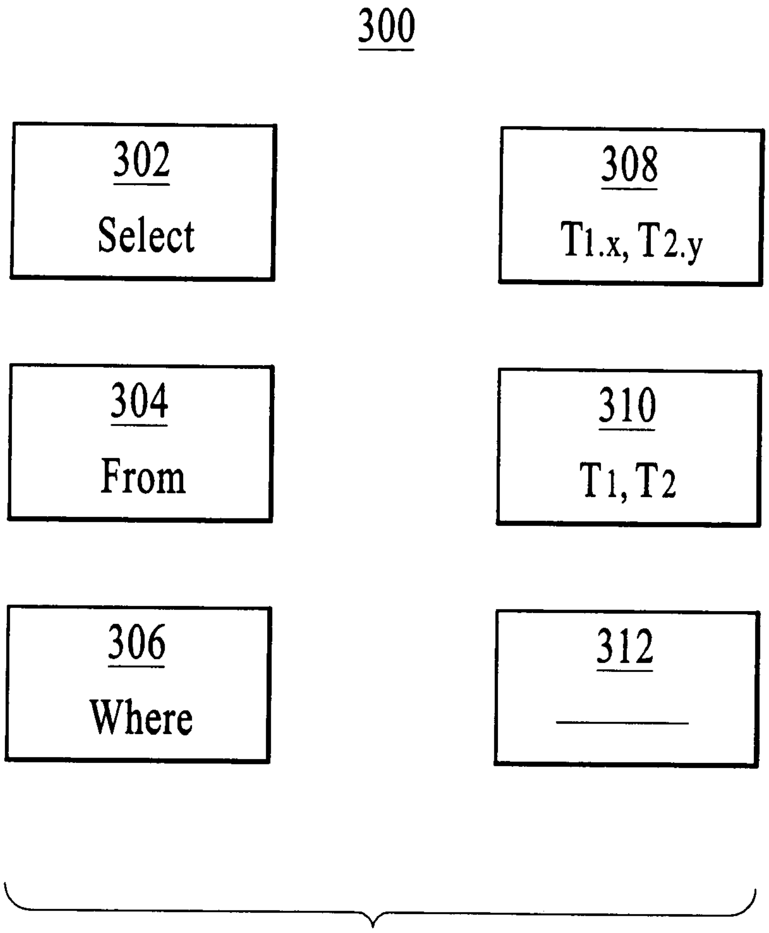


FIG. 3

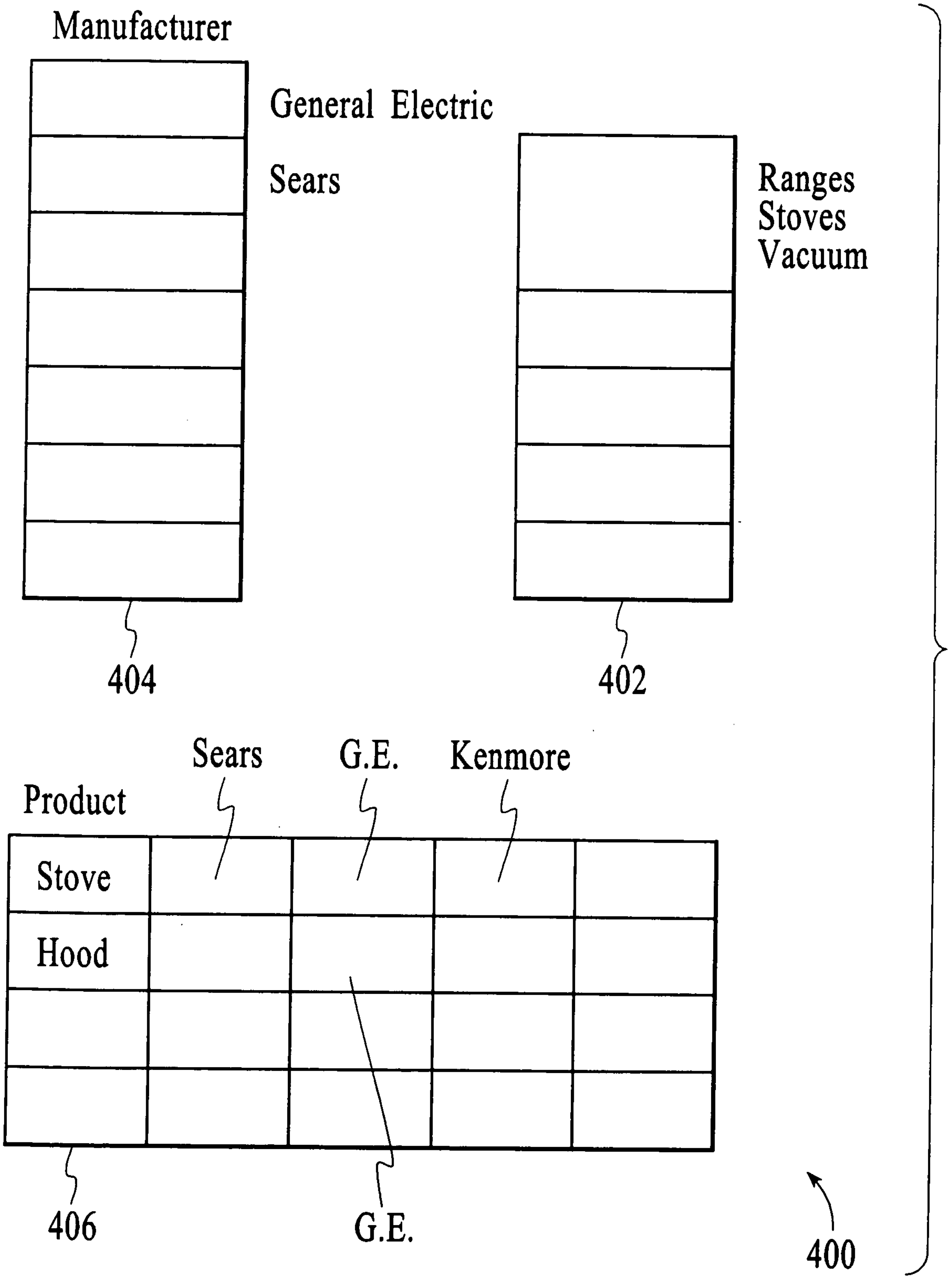
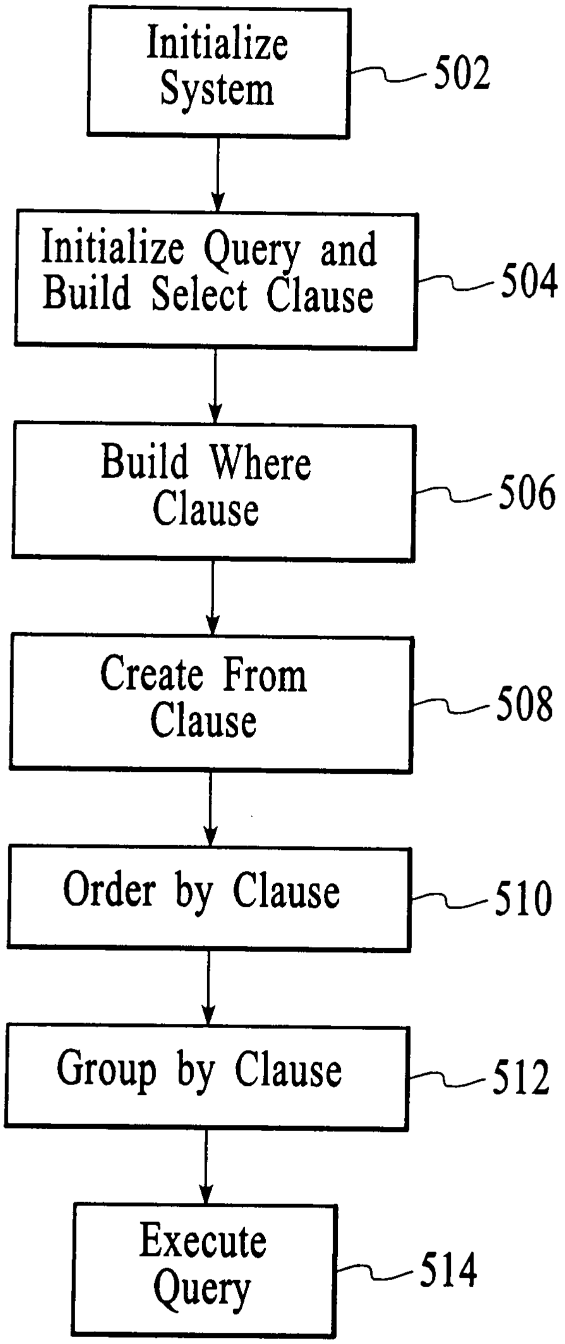
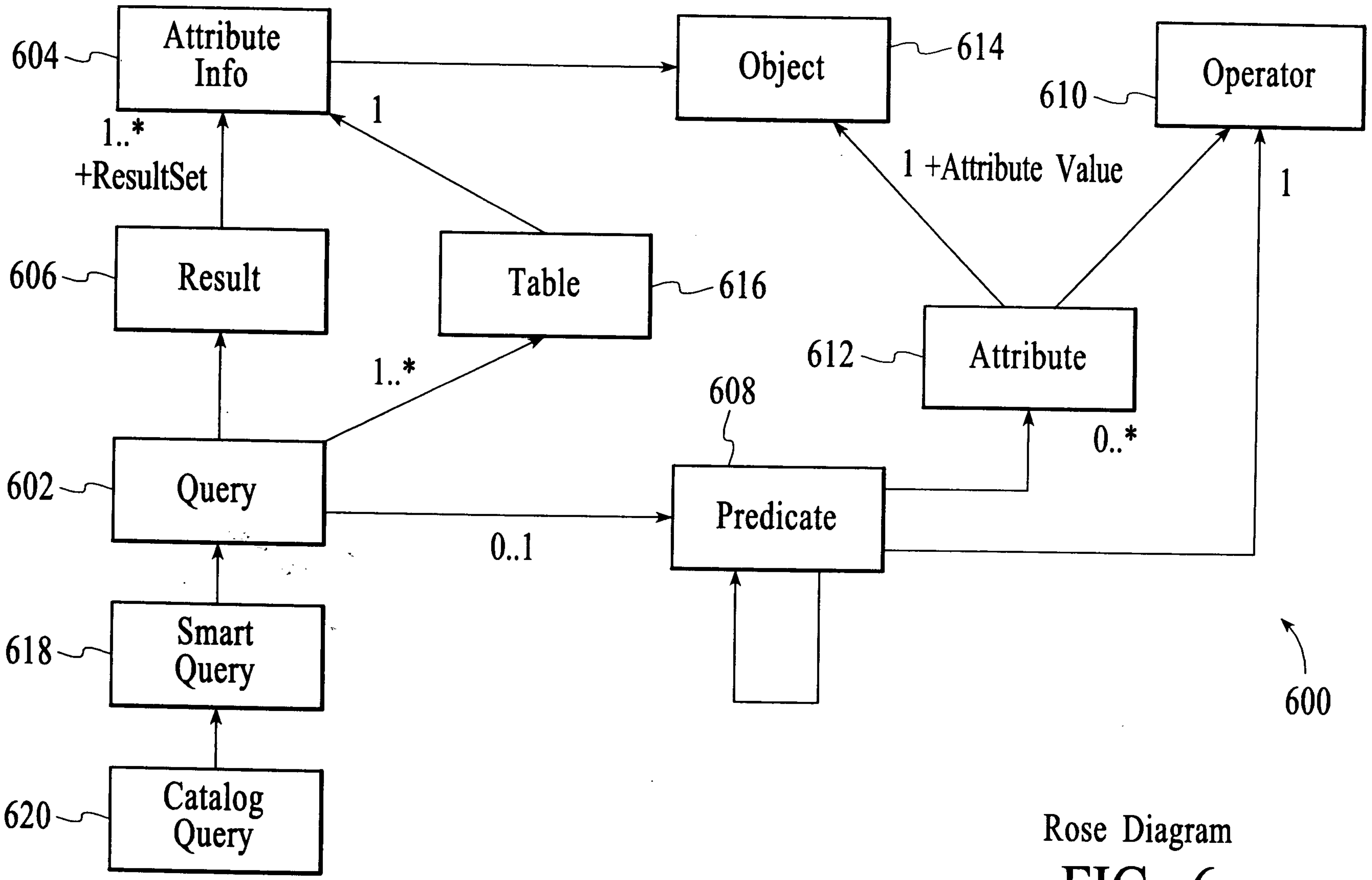

Manufacturer
General Electric
Sears
Ranges
Stoves
Vacuum
404
402
Sears
G.E.
Kenmore
Product
Stove
Hood
406
G.E.
400

FIG. 4


Initialize System
502
Initialize Query and Build Select Clause
504
Build Where Clause
506
Create From Clause
508
Order by Clause
510
Group by Clause
512
Execute Query
514

500

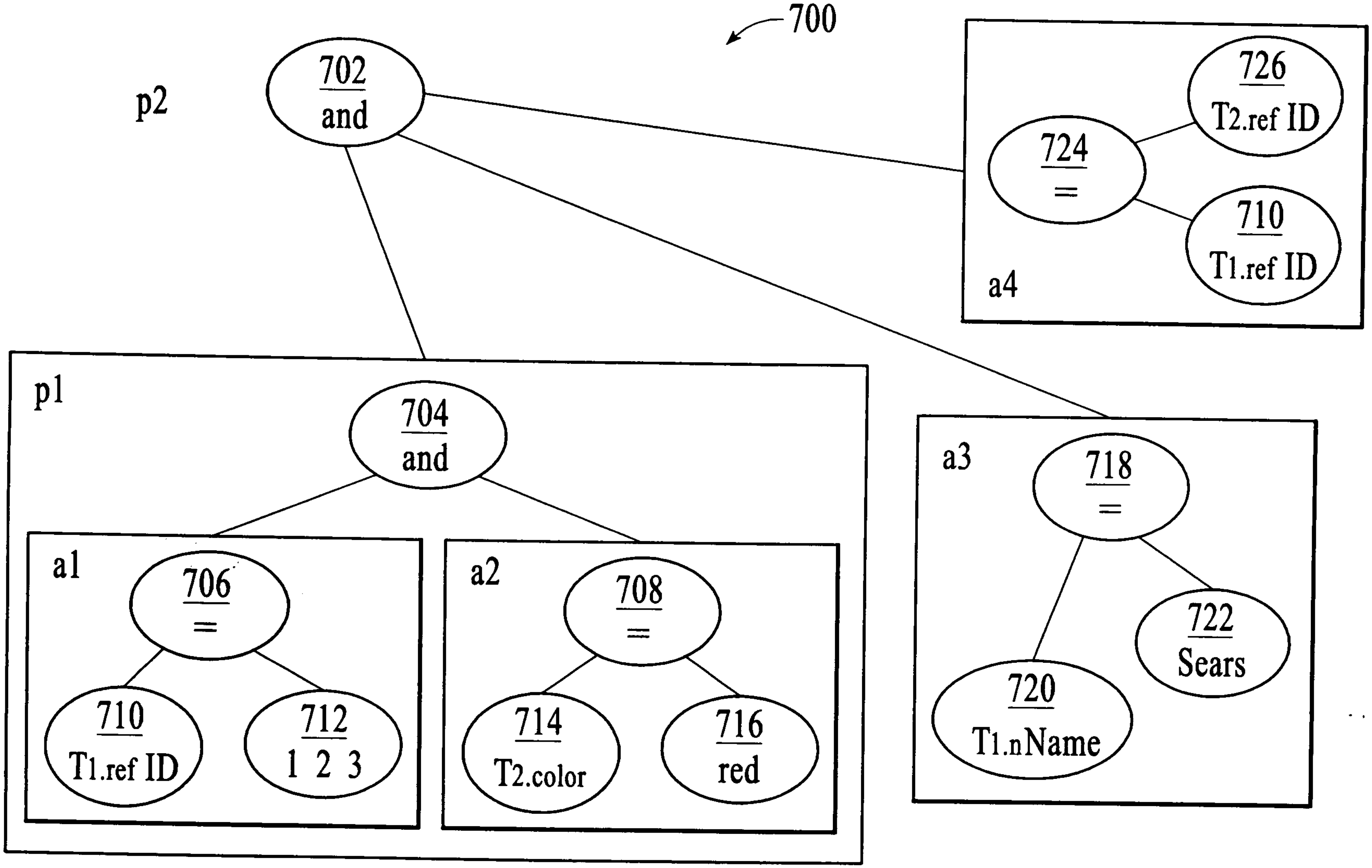
FIG. 5



Rose Diagram

FIG. 6

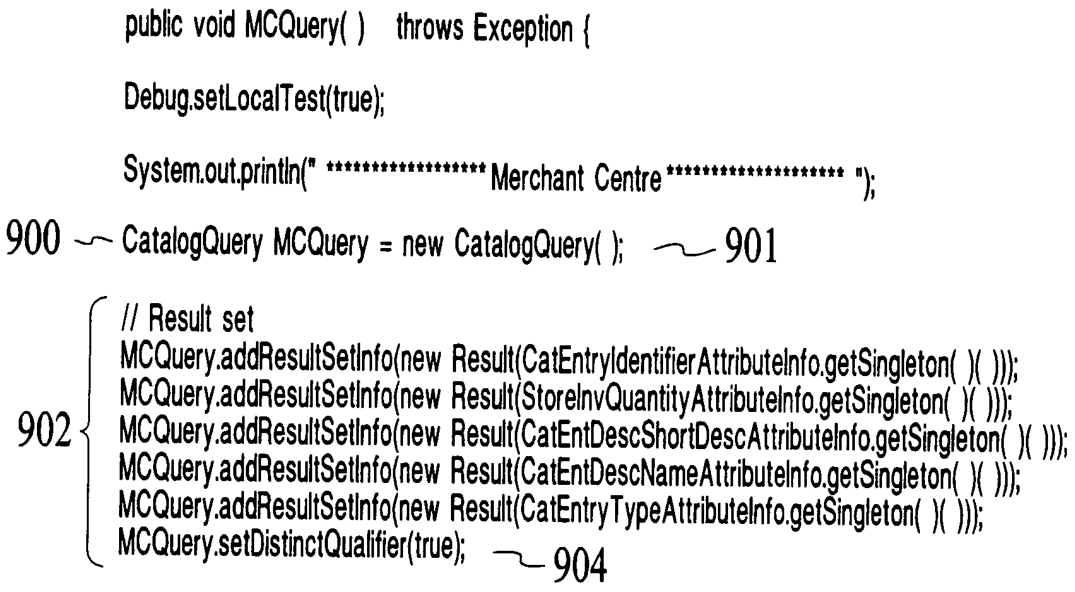
FIG. 7


700
p2
702
and
a4
724
=
726
T2.ref ID
710
T1.ref ID
p1
704
and
a1
706
=
710
T1.ref ID
712
1 2 3
a2
708
=
714
T2.color
716
red
a3
718
=
720
T1.nName
722
Sears

```
    ┌ // construct the simple search conditions
    │ Attribute attr1 = new Attribute (CatRefIdAttributeInfo, Operator.equal, "123");
    │ Attribute attr2 = new Attribute (Colourttributelnfo, Operator.equal, "red");
802 ┤ Attribute attr3 = new Attribute (ManufactureAttributeInfo, Operator.equal, "Sears");
    │ Attribute attr4 = new
    └ Attribute (CatRefIdAttributeInfo,Operator,equal,DescRefIdAttributeInfo,);

    ┌ // compose composite search conditions
804 ┤ Predicate p1 = new Predicate (Operator.and, {attr1, attr2} );
    └ Predicate p2 = new Predicate (Operator.and, {p1, attr3, attr4} );

    ┌ // execute the query
806 ┤ Query q = new Query ( )
    │ q.setResultSet ({ CatRefIdAttributeInfo, ...}) // result set contains catalog entryId
    └ q.setPredicate (p2);
808 ~ result = q.execute ( )
```

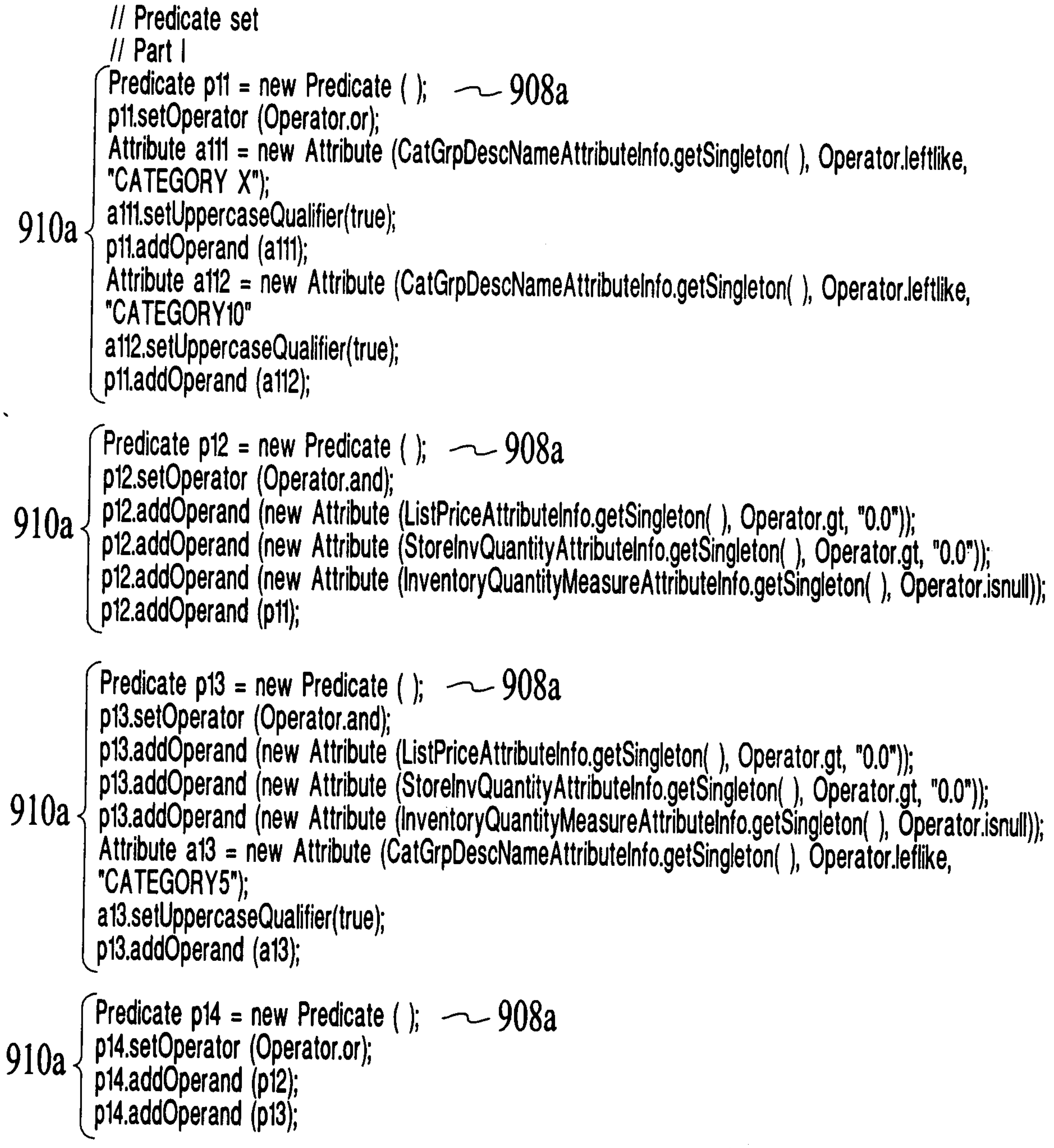
800

FIG. 8



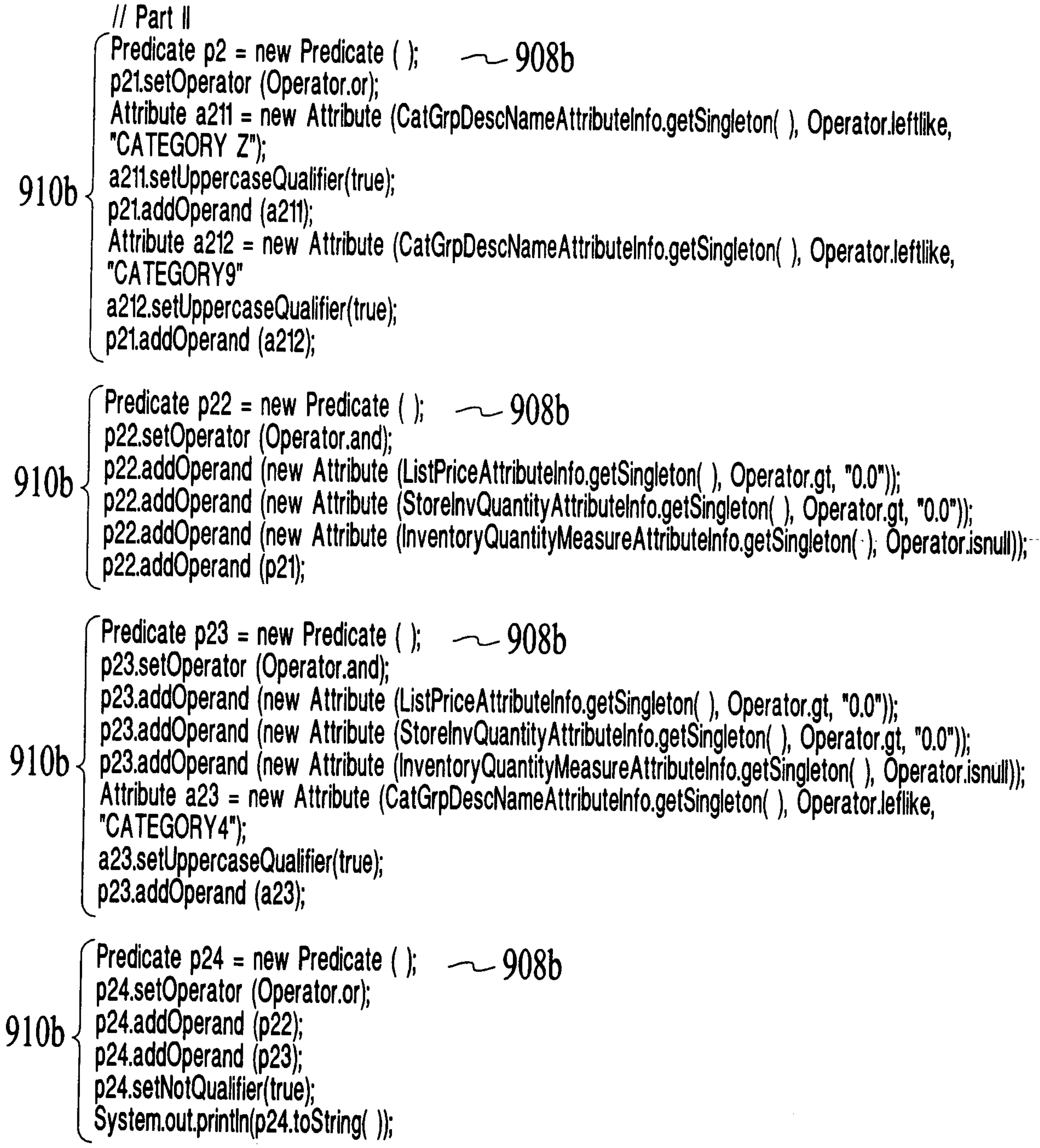
```
public void MCQuery( )   throws Exception {

Debug.setLocalTest(true);

System.out.println(" ****************** Merchant Centre ******************** ");

CatalogQuery MCQuery = new CatalogQuery( );

// Result set
MCQuery.addResultSetInfo(new Result(CatEntryIdentifierAttributeInfo.getSingleton( )( )));
MCQuery.addResultSetInfo(new Result(StoreInvQuantityAttributeInfo.getSingleton( )( )));
MCQuery.addResultSetInfo(new Result(CatEntDescShortDescAttributeInfo.getSingleton( )( )));
MCQuery.addResultSetInfo(new Result(CatEntDescNameAttributeInfo.getSingleton( )( )));
MCQuery.addResultSetInfo(new Result(CatEntryTypeAttributeInfo.getSingleton( )( )));
MCQuery.setDistinctQualifier(true);
```

FIG. 9A(i)



```
// Predicate set
// Part I
Predicate p11 = new Predicate ( );
p11.setOperator (Operator.or);
Attribute a111 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
"CATEGORY X");
a111.setUppercaseQualifier(true);
p11.addOperand (a111);
Attribute a112 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
"CATEGORY10"
a112.setUppercaseQualifier(true);
p11.addOperand (a112);

Predicate p12 = new Predicate ( );
p12.setOperator (Operator.and);
p12.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p12.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p12.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
p12.addOperand (p11);

Predicate p13 = new Predicate ( );
p13.setOperator (Operator.and);
p13.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p13.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p13.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
Attribute a13 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leflike,
"CATEGORY5");
a13.setUppercaseQualifier(true);
p13.addOperand (a13);

Predicate p14 = new Predicate ( );
p14.setOperator (Operator.or);
p14.addOperand (p12);
p14.addOperand (p13);
```

FIG. 9A(ii)



```
// Part II
Predicate p2 = new Predicate ( );
p21.setOperator (Operator.or);
Attribute a211 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
"CATEGORY Z");
a211.setUppercaseQualifier(true);
p21.addOperand (a211);
Attribute a212 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leftlike,
"CATEGORY9"
a212.setUppercaseQualifier(true);
p21.addOperand (a212);

Predicate p22 = new Predicate ( );
p22.setOperator (Operator.and);
p22.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p22.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p22.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
p22.addOperand (p21);

Predicate p23 = new Predicate ( );
p23.setOperator (Operator.and);
p23.addOperand (new Attribute (ListPriceAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p23.addOperand (new Attribute (StoreInvQuantityAttributeInfo.getSingleton( ), Operator.gt, "0.0"));
p23.addOperand (new Attribute (InventoryQuantityMeasureAttributeInfo.getSingleton( ), Operator.isnull));
Attribute a23 = new Attribute (CatGrpDescNameAttributeInfo.getSingleton( ), Operator.leflike,
"CATEGORY4");
a23.setUppercaseQualifier(true);
p23.addOperand (a23);

Predicate p24 = new Predicate ( );
p24.setOperator (Operator.or);
p24.addOperand (p22);
p24.addOperand (p23);
p24.setNotQualifier(true);
System.out.println(p24.toString( ));
```

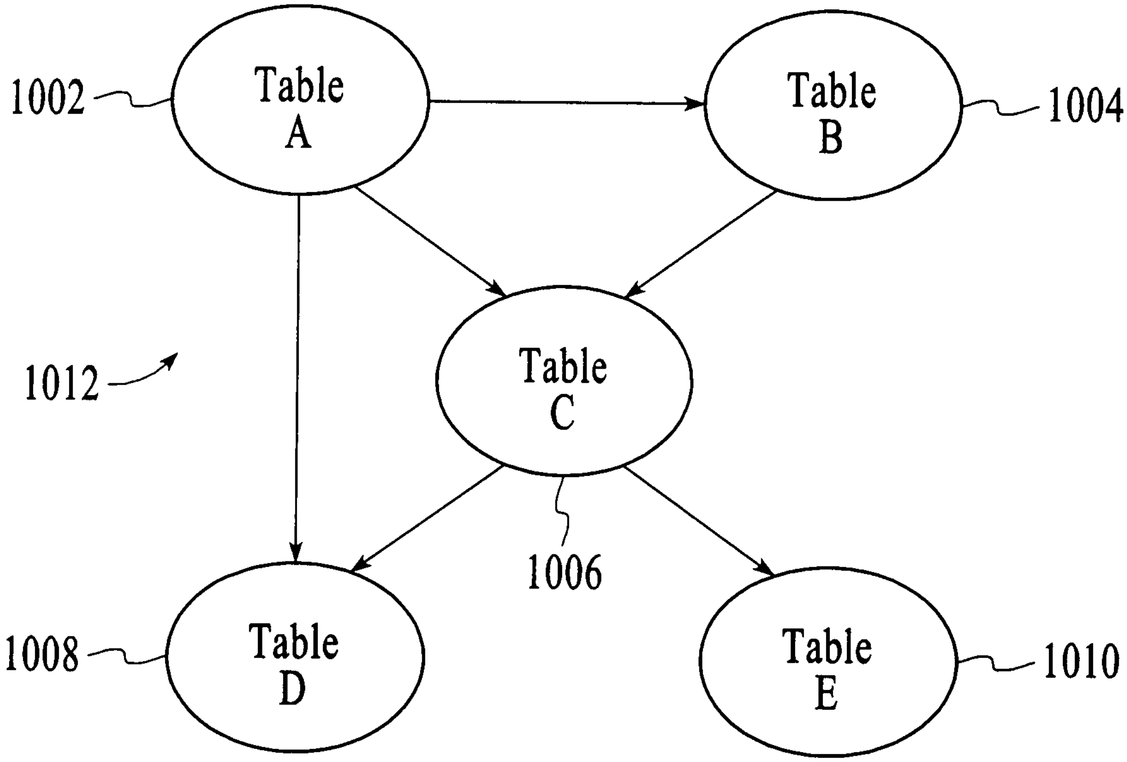
FIG. 9A(iii)

```
// Part IV -- Join
Predicate p4 = new Predicate ( );    912
p4.setOperator (Operator.and);
p4.addOperand (p14);
p4.addOperand (p24);
p4.addOperand (new Attribute (StoreCEntStoreIdentifierAttributeInfo.getSingleton( ),    914
Operator.eq, "2"));
p4.addOperand (new Attribute (UsersIdentifierAttributeInfo.getSingleton( ), Operator.eq,
"1001"));
//p4.addOperand (p33);

MCQuery.setPredicate(p4);    916

// Join
System.out.println("Auto Join : ");
MCQuery.printJointRelationships( );

// Resolve source tables
MCQuery.resolveSourceTables( );    918

// ORDER, GROUP and HAVING set
MCQuery.addResultOrder(CatEntryIdentifierAttributeInfo.getSingleton( ),
Operator.desc);    920

System.out.println("MC Query : ");
System.out.println(MCQuery.toString( ));

com.ibm.commerce.base.objects.Cursor cursor = new
com.ibm.commerce.base.objects.Cursor( );
java.util.Vector v = MCQuery.execute(cursor);    922
System.out.println("MC Query first 10 Result: ")
System.out.println(v);
cursor.increment( );
v = MCQuery.execute(cursor);    922
System.out.println("MC Query next 10 Result: ");
System.println(v);

}
```

FIG. 9A(iv)

```
public void setPredicate(Predicate aPredicate) throws Exception {

    Predicate additionalP = additionalPredicate( );    924
    if (additionalP != null) {
      Predicate p = new Predicate( );
926   p.setOperator(Operator.and);
      p.addOperand(aPredicate);
      p.addOperand(additionalP);
      setTableJointPredicate(p);
    }
    else
      setTableJointPredicate(aPredicate);
    }


private void setTablejointPredicate(Predicate aPredicate) throws Exception {

Predicate jointP = resolveJointPredicate(aPredicate);
if (jointP != null) {
  Predicate p = new Predicate( );
  p.setOperator(Operator.and);
  p.addOperand(aPredicate);
  p.addOperand(jointP);
  super.setPredicate(p);
}                                930
else
  super.setPredicate(aPredicate);
}
```

FIG. 9B


1002
Table A
Table B
1004
1012
Table C
1006
1008
Table D
Table E
1010

1000

FIG. 10

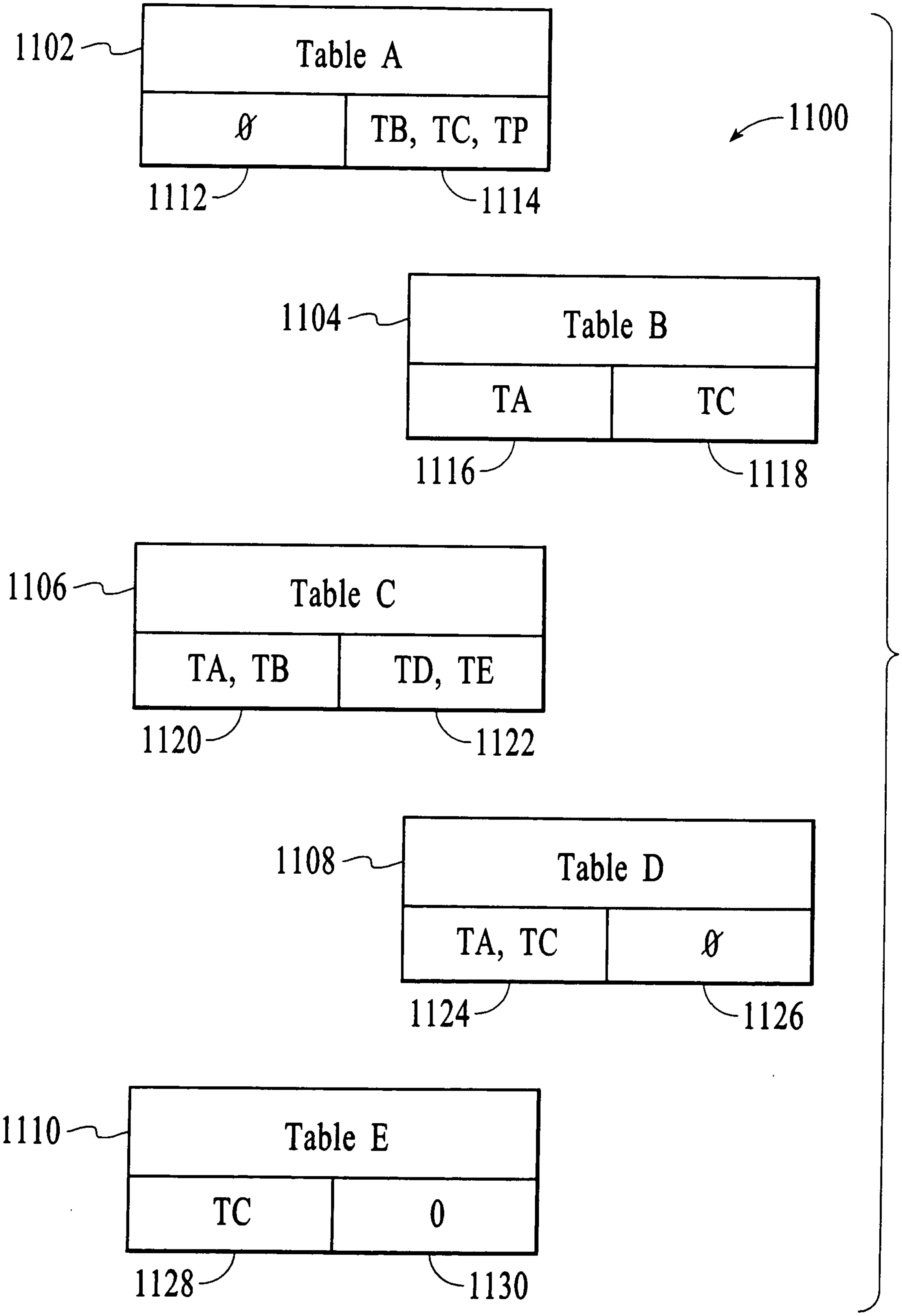
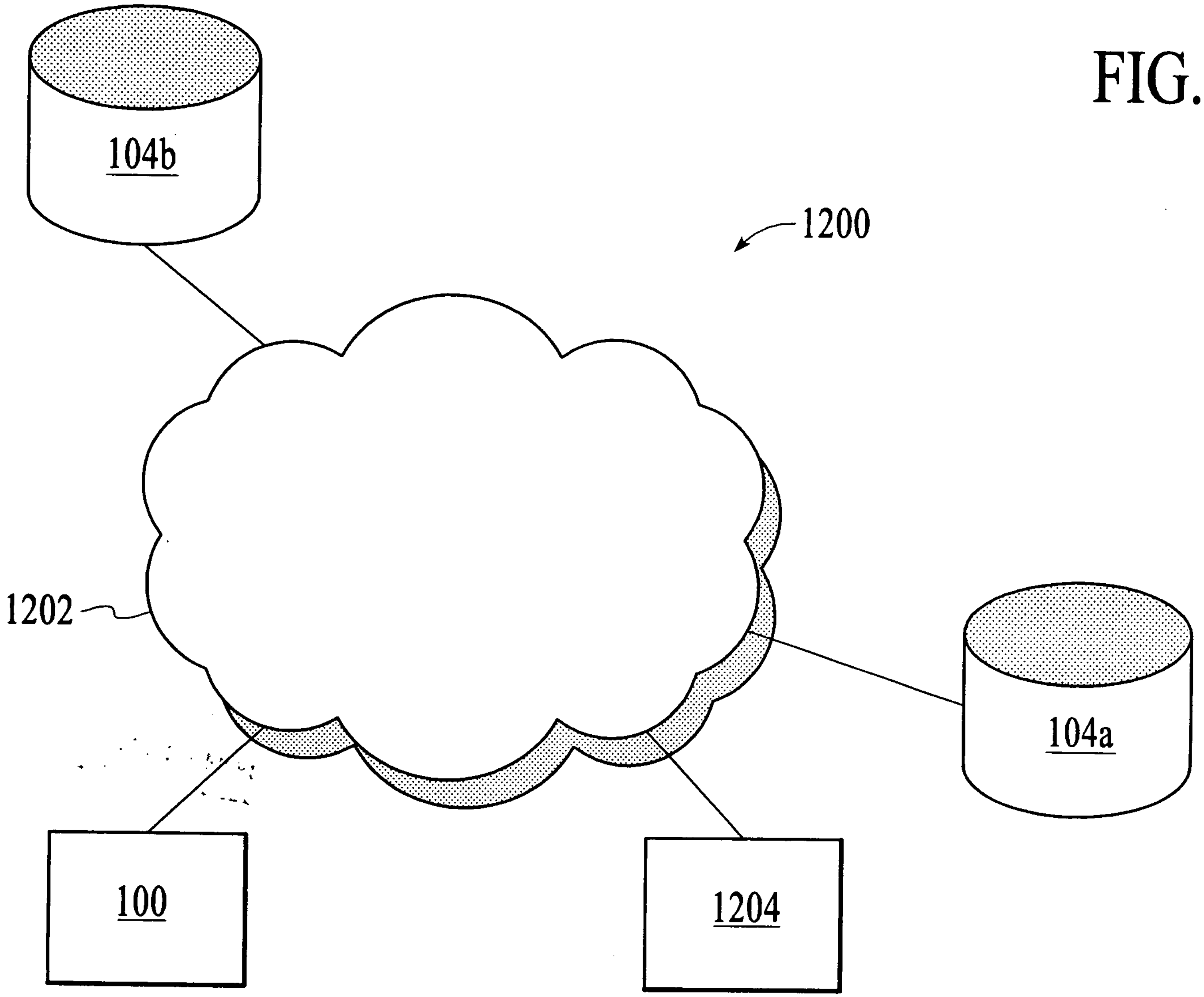

1100
1102
Table A
Ø
TB, TC, TP
1112
1114
1104
Table B
TA
TC
1116
1118
1106
Table C
TA, TB
TD, TE
1120
1122
1108
Table D
TA, TC
Ø
1124
1126
1110
Table E
TC
0
1128
1130

FIG. 11

FIG. 12


104b
1200
1202
104a
100
1204